Merlin ToolChain

Enabling transparency beyond partner borders

Rob Kommeren - Philips

2007-09-25

Content

1. Background

2. Merlin Toolchain
   - Goals
   - Architecture
   - Implementation

3. Implementation – The Demonstrator

4. Implementation – Philips OSIB project

5. Conclusions, current and future work

Background

- With small scale product development of the past, only few tools were used.
- Nowadays software development is becoming increasingly complex and productivity requirements are so high that development tools are indispensable.
- No tool or development entity is an island. Instead, they are connected and these connections need to be managed efficiently.
- Interoperability is not readily available in development tools. Data transfer between tools used in different SW development phases is considered difficult.
- Tools from different vendors do not share common data formats. As a result manual data transferring and handling is needed
  - Manual data transfer is laborious and increases errors.
- The impact of the issues stated above is even higher in view of the trend to collaborative software development…

Collaborative Perspective

- Companies rarely have the same sets of software tools. Typical consequent problems are:
  - Poor coordination and synchronisation of work flow
  - Poorly alignment of work products and their statuses
  - Lack of transparency of development for parties involved

  … Resulting in an inefficient and ineffective development process

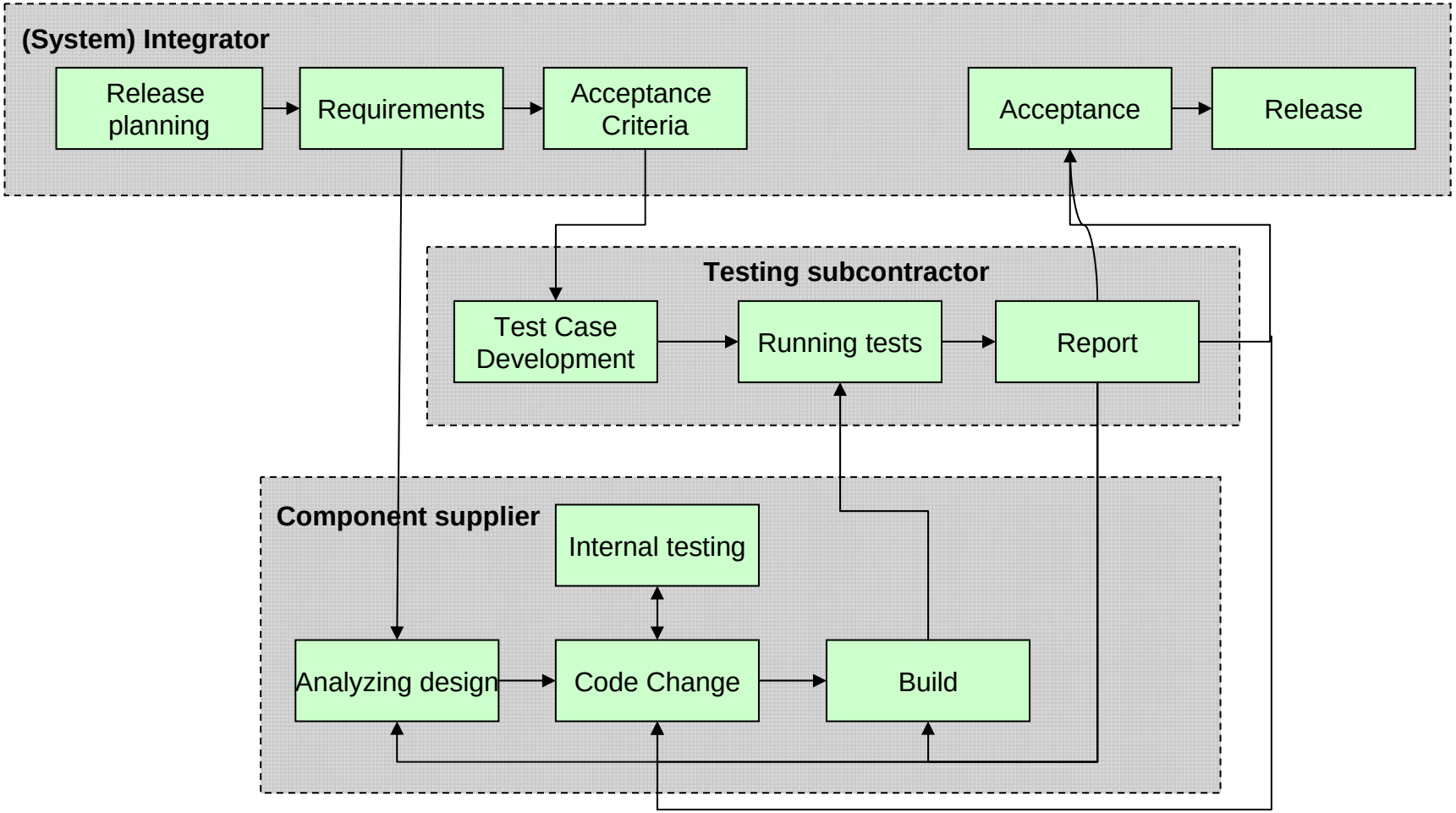
- Constraint:
  Industrial experience shows the need for integrable rather than integrated tool solutions
  - Proven solutions for parts of the problems should be kept!

The solution**: Tool integration optimised for personal needs!**

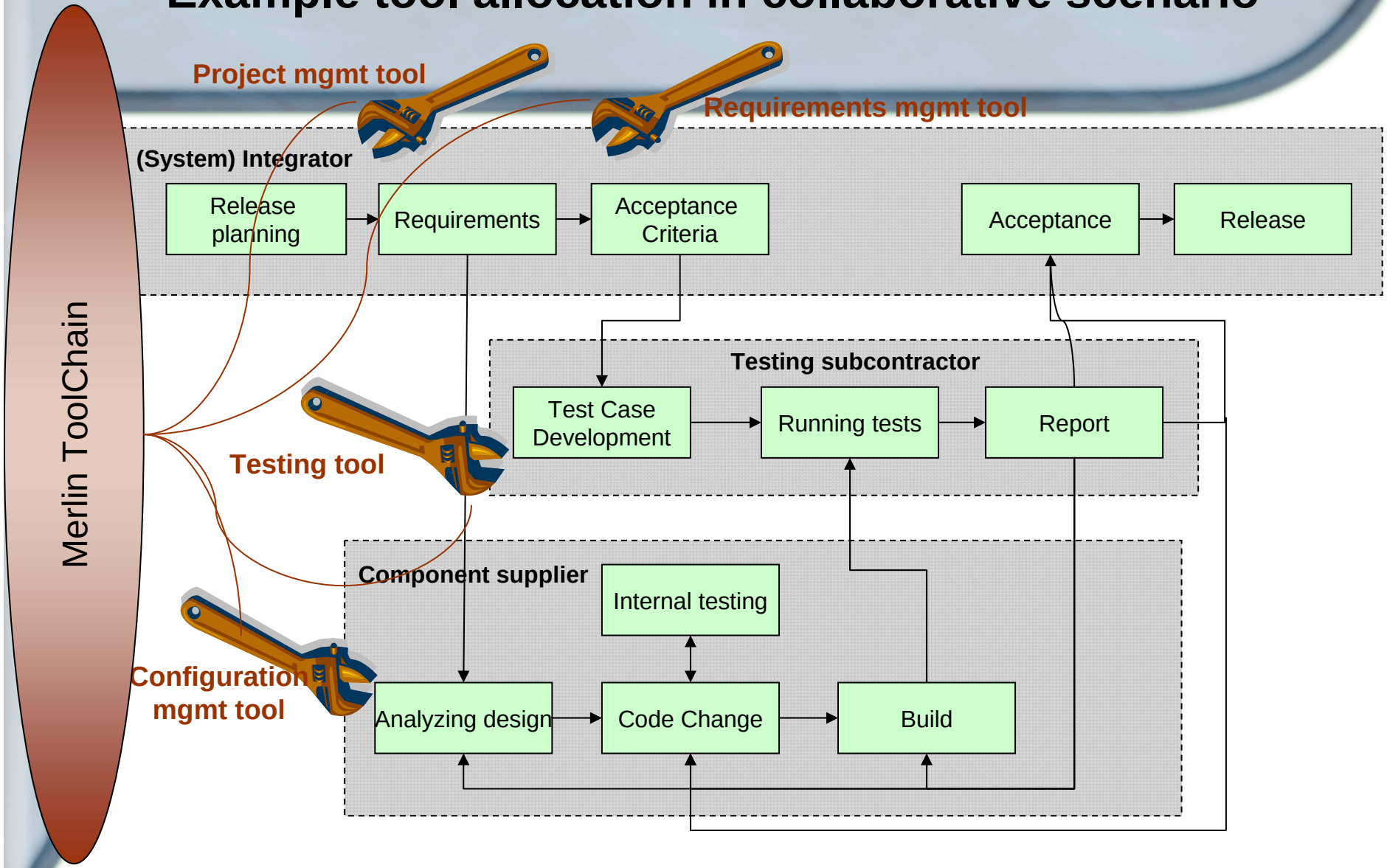
Merlin ToolChain Goals

- To support collaborative distributed SW projects by developing a concept for integrating existing tools in a meaningful way
- To build an example implementation of the Merlin Toolchain by using selected sets of tools
- To pilot the implementation both in a demonstrator and in a real life industrial project
- To gather experiences for further development

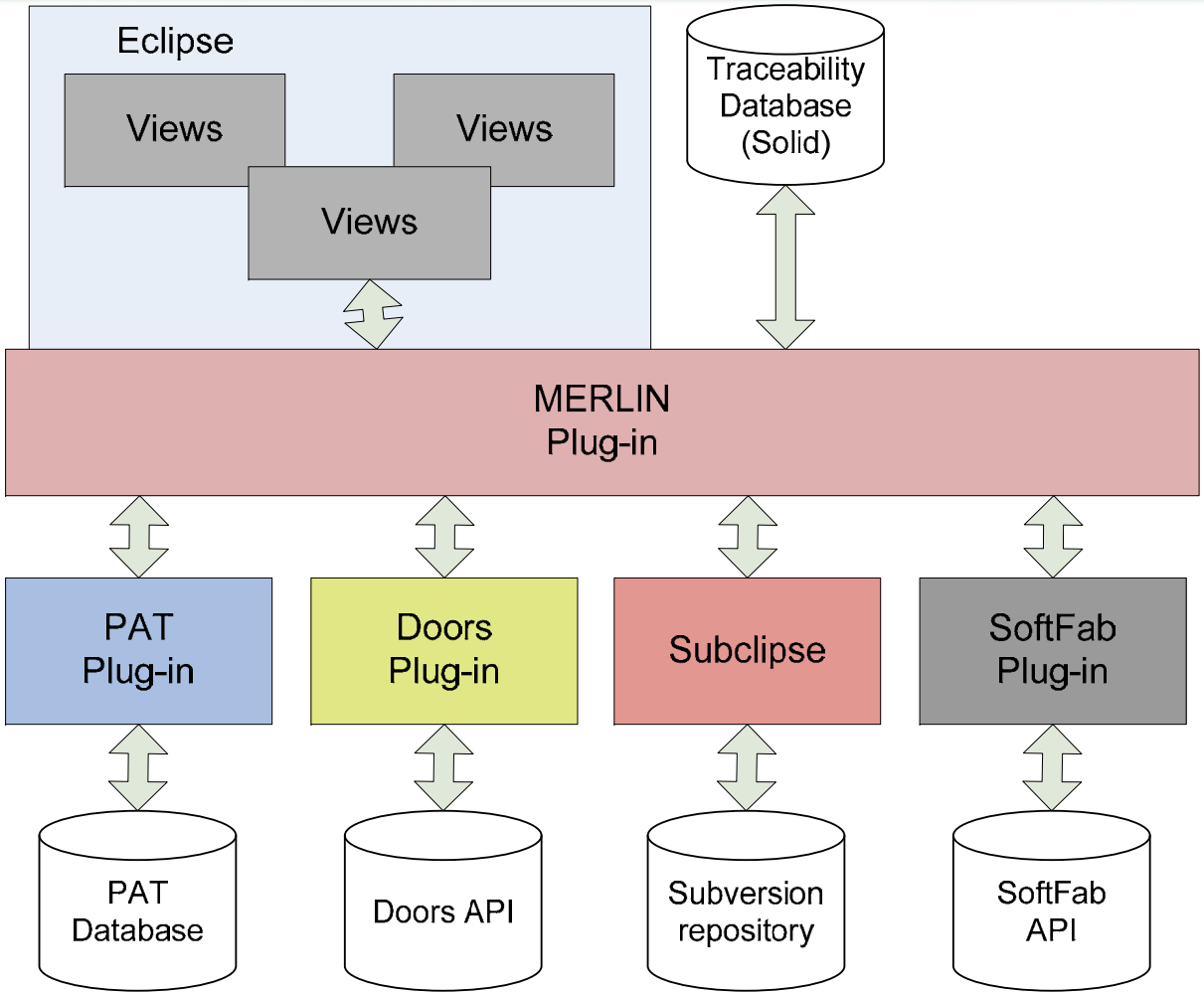
Reference: Typical collaborative scenario



Example tool allocation in collaborative scenario



Merlin ToolChain Overall Architecture

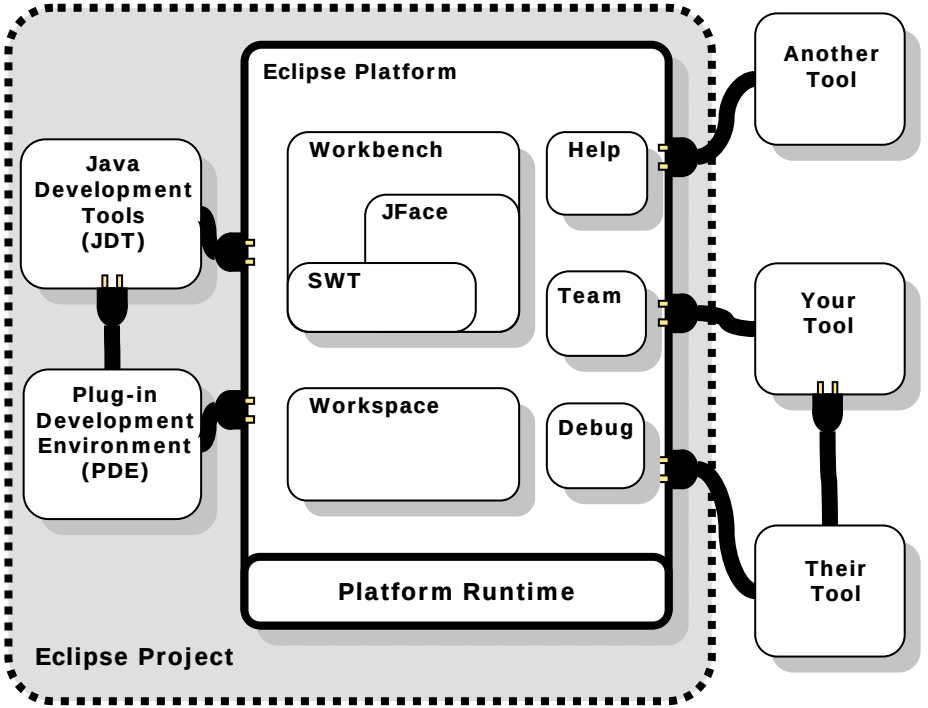


Tool Chain Implementation

- Eclipse is used as an integration framework
- The ToolChain is implemented by developing Eclipse plug-ins:
  - The main plug-in called Merlin Plug-in controls these separate plug-ins and formulates the completeness of the tool chain under the Eclipse IDE
  - Tools are accessed with own separate plug-ins. E.g.,:
    - Doors (made for Telelogic Doors tool)
    - PATClipse (made for Philips PAT tool)
    - Subclipse (Existing plug-in, integrated with ToolChain)
    - SoftFab (made for Philips SoftFab tool)

Eclipse Introduction

- The Eclipse Platform is an open source tool designed for building integrated development environments (IDEs)
- Can be used to develop e.g. Java and C++ programs, supporting also modelling, testing and embedded development
- Written in the Java language and comes with plug-in based framework that makes it easier to create, integrate and utilize software tools
- Eclipse Platform has a small kernel called as the Platform Runtime, but almost all functionality is located in plug-ins
- A plug-in is the smallest unit of Eclipse Platform function that can be developed and delivered separately



Selection criteria for tools to be incorporated in the ToolChain

- To start, for each of the following development areas one tool was selected
  - Project Management
  - Requirement Management
  - Configuration Management
  - Testing
- Selection of the was based on investigation at Merlin industrial partners
  - Tools that some industrial partners are using
  - Tools that were wanted for detailed analysis
- Tools were also selected from different vendors on purpose, to emulate the real world situation as well as possible

Selected Tools

- Initially selected tools were
  - **Philips Project Assist Tool** (Project Management)
  - **IBM Rational RequisitePro** (Requirements Management)
  - **Telelogic Synergy/CM** (Configuration Management)
  - **Philips SoftFab** (Testing)
- To prove that tools could be exchanged in the ToolChain, new tools were introduced in a later stage:
  - **RequisitePro → Telelogic Doors™**
  - **Telelogic Synergy/CM → Subversion**

Merlin ToolChain is customizeable!

Use any combination or subset of the tools already plugged in

Or, plug-in your own tool according to instructions!

OPEN WORKBENCH
Open-Source Project Scheduling

PHILIPS
Project Assist Tool

Project mgmt tool

SynergyCM

SUBVERSION
Subversion

Rational RequisitePro

Doors

OSRMT

Requirements mgmt tool

Test Link

SoftFab

Demonstrator case

Three partners are working in collaboration

- Integrator (Finland)
  - Requirements Management and Project management
- Supplier (USA)
  - Software implementation
- Subcontractor (China)
  - Software testing

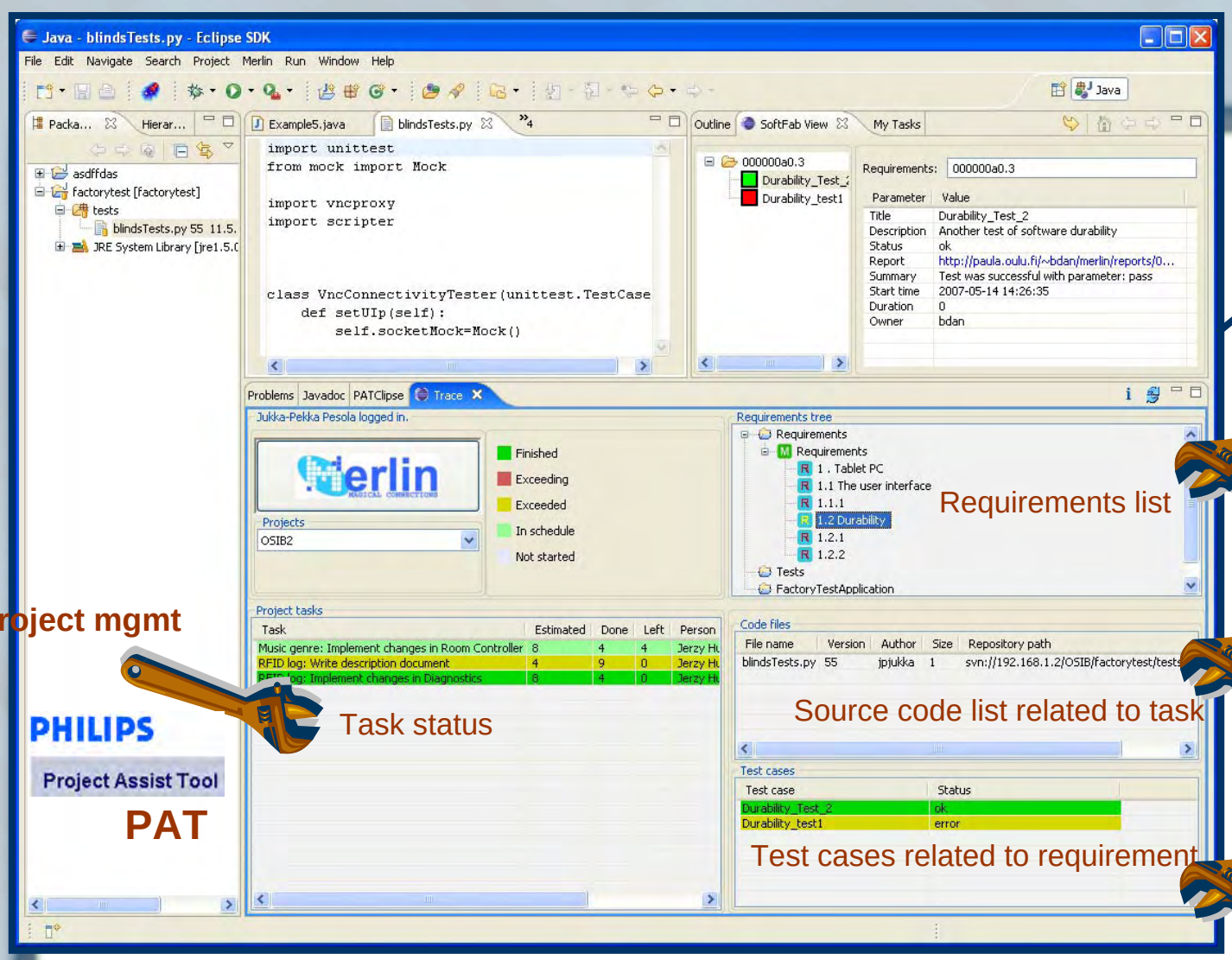
Demonstrator Goals

- Uniform project view for all partners
- Up-to-date information available to all the partners

Demonstrator script

- Integrator fills in tasks to project mgmt tool and assigns them to subcontractor and supplier
  - Tasks can be monitored in Merlin ToolChain by all parties
- Integrator writes requirement description in RM tool and links it to tasks in the Merlin ToolChain
- Integrator fills effort estimation for the task
- Component supplier sees new task assignment and develops new code accordingly
- Component supplier records effort spent in project mgt tool
- Testing subcontractor sees component supplier is ready with code development
- Testing subcontractor runs tests for the component
- Testing subcontractor records effort spent
- Integrator can see the test results and accepts the component accordingly

Actual implementation



Requirements list

Task status

Source code list related to task

Test cases related to requirement

Requirements mgmt

DOORS

Configuration mgmt

Subversion

Project mgmt

PHILIPS

Project Assist Tool

PAT

Testing

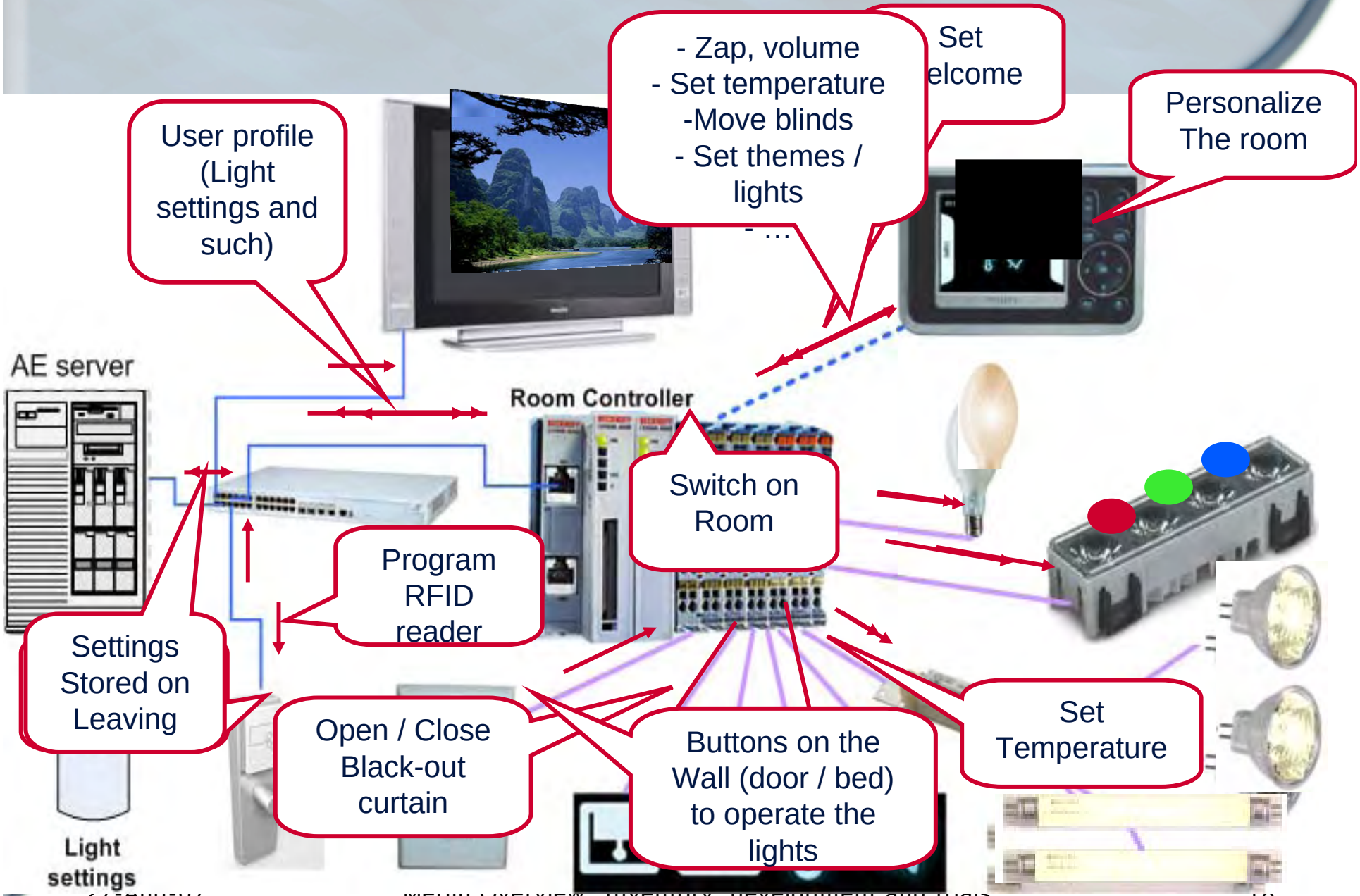
SoftFab

Industrial case: Philips OSIB project

Purpose of the industrial case

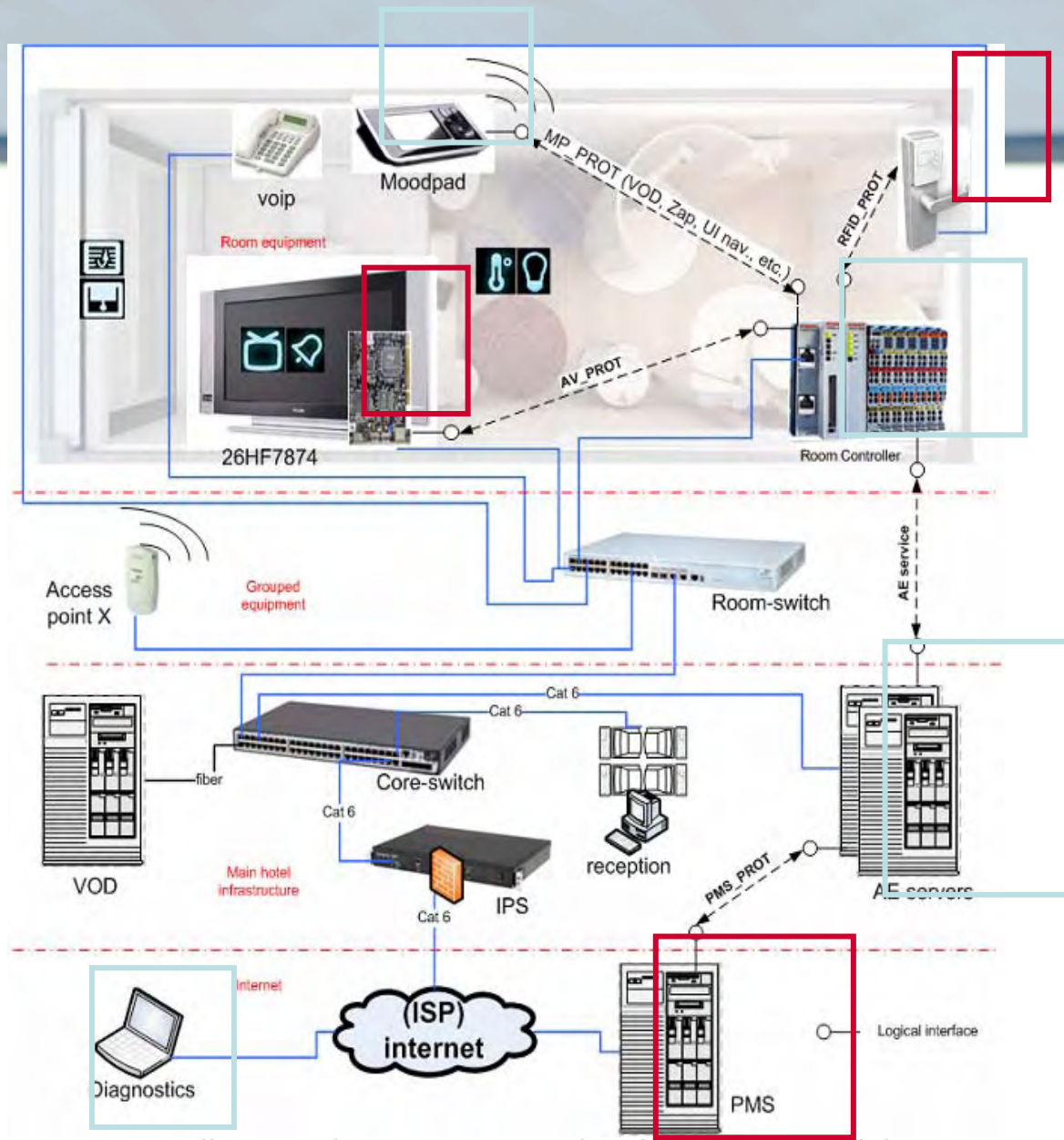
- To prove that Merlin solutions work in an integrated fashion in real-life industrial projects (referring ITEA reviewers' comments)
- To implement the Merlin tool chain in an industrial project
  - Enabling smooth continuation of the day-to-day project execution
  - Targeted added value for the project:
    Full transparency and traceability of
    - Project tasks
    - Requirements
    - Related source code
    - Build and test status

MerlinThe OSIB project: Today's Experience



The OSIB project: Hospitality implementation

3rd party integration points

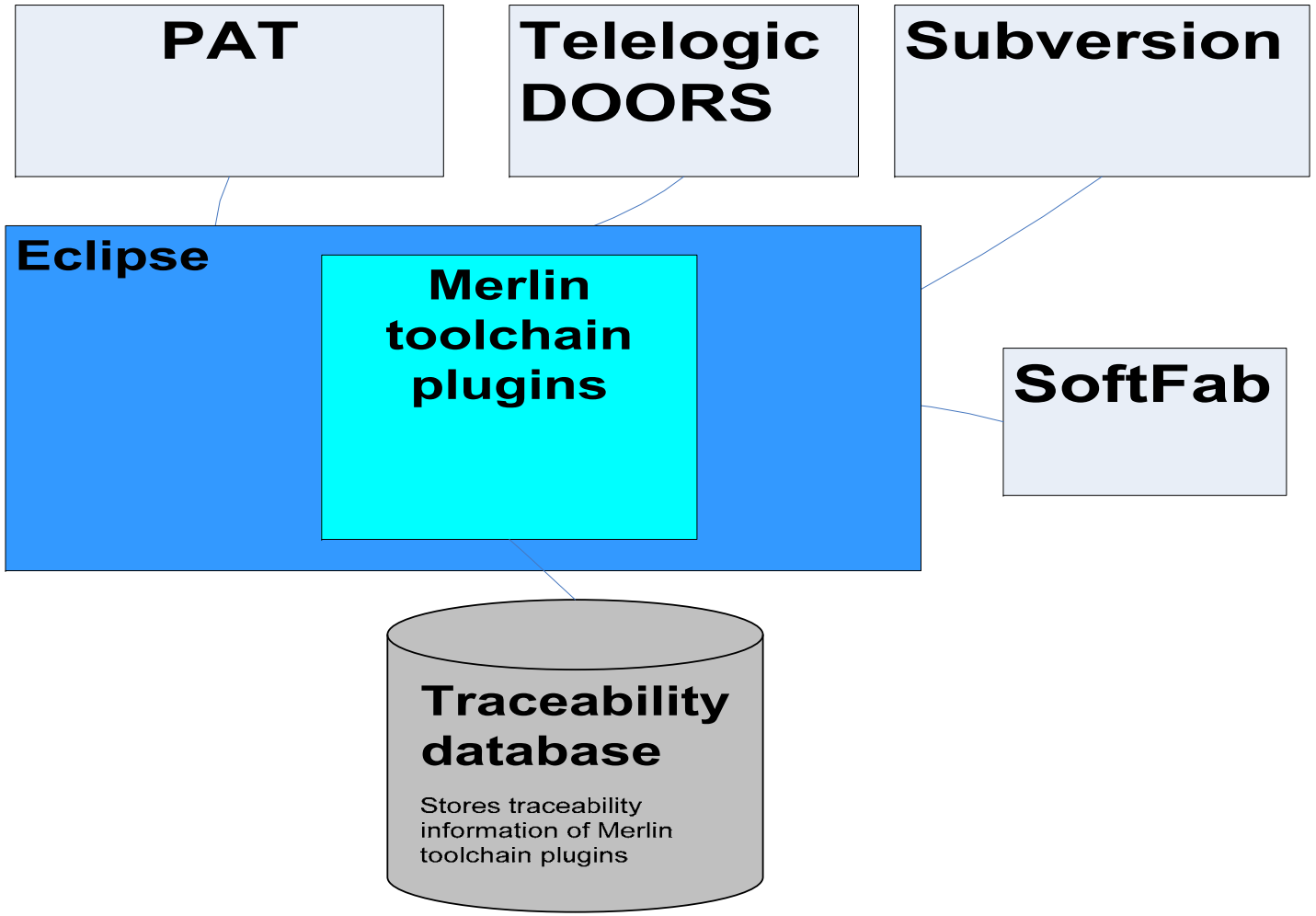


Software made by Philips

Project characteristics

- Implementation agreed with customer in phases of 3-6 months
  - Project is in 3rd implementation phase
- Team size: 6 engineers (during implementation)
- Agile way of working
  - Development in increments of two weeks
  - Requirements are selected for each increment, together with customer
- OSIB's native tool-set: PAT, DOORS, Subversion, SoftFab

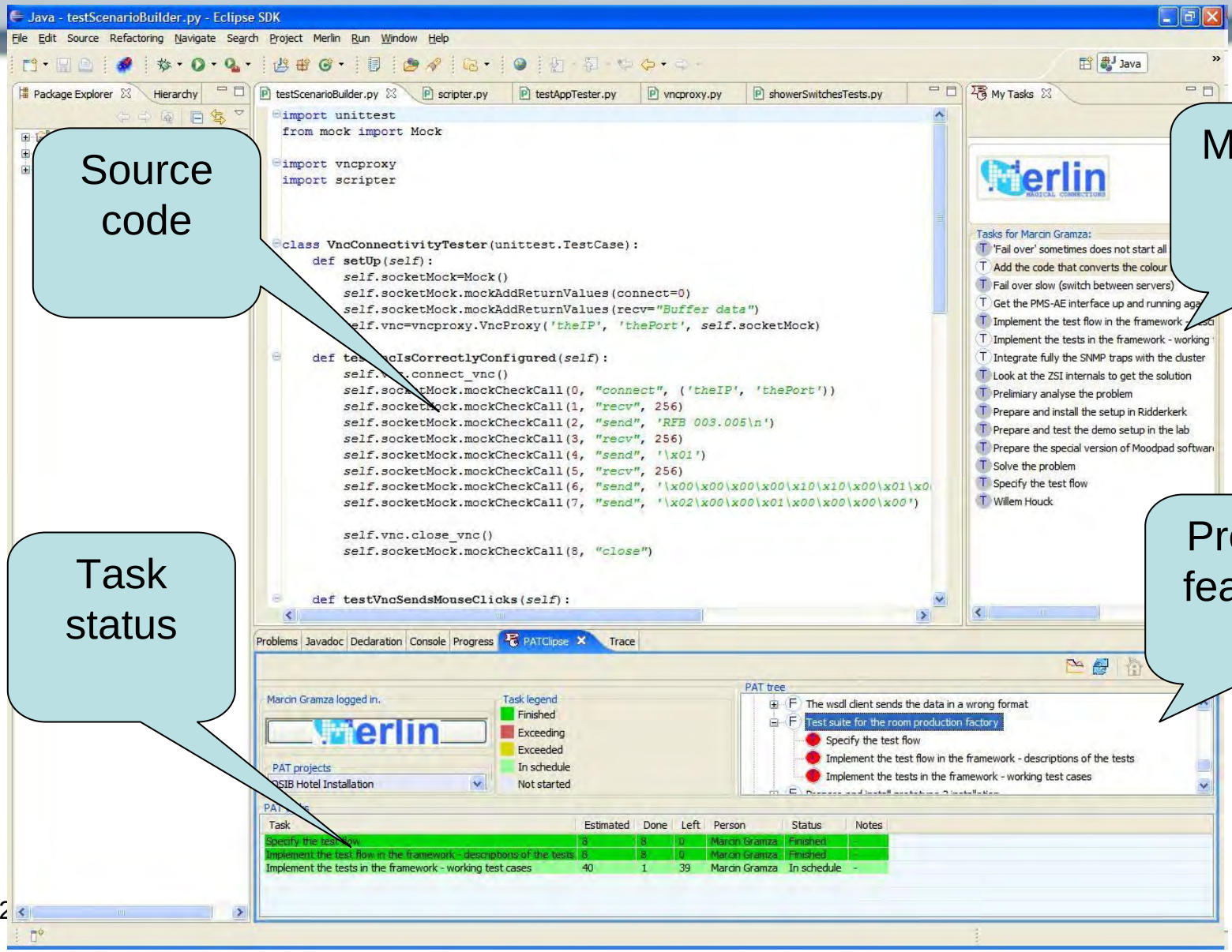
Merlin ToolChain: Instantiation for OSIB



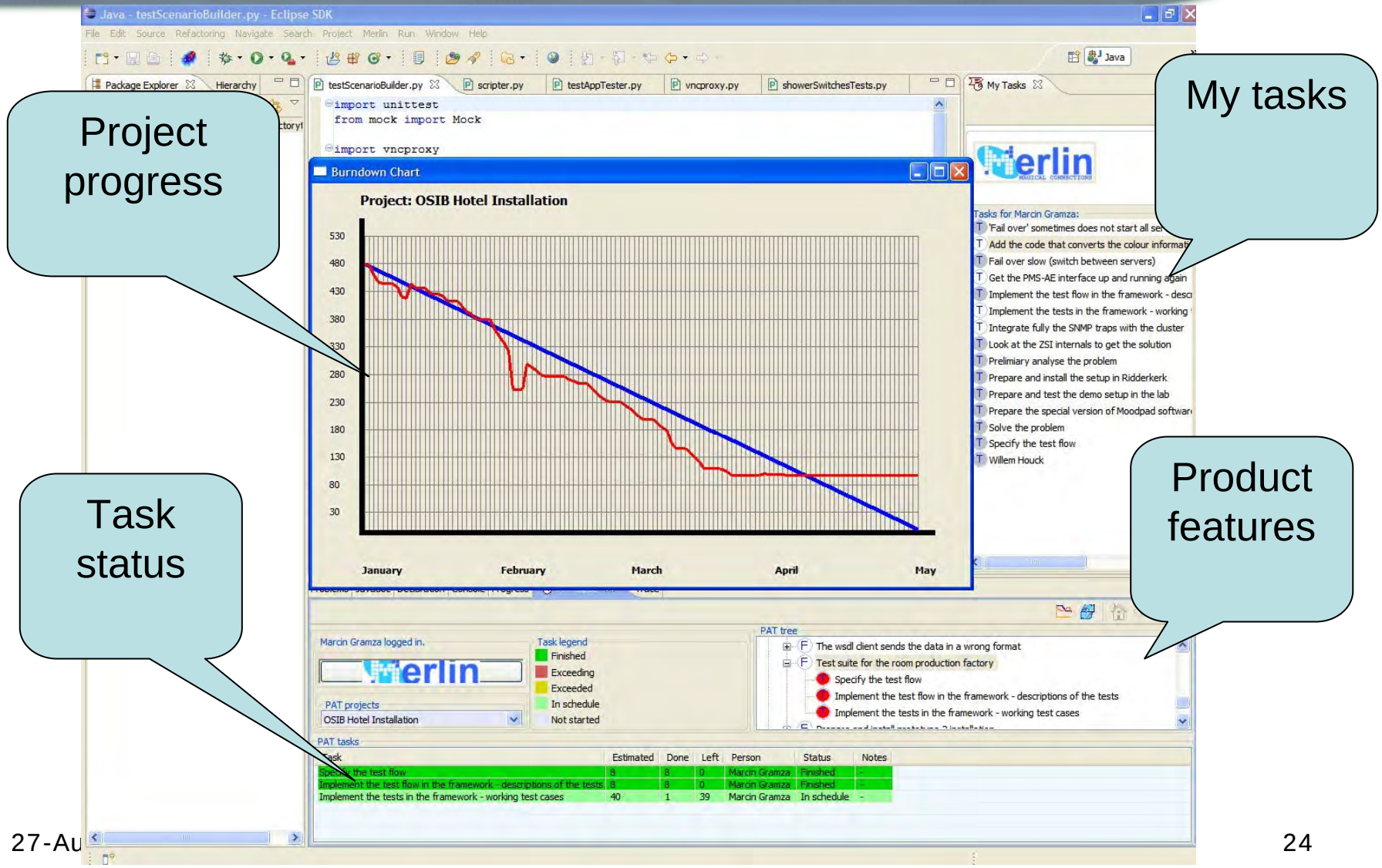
Implementation trajectory

Wk 710-712	Preparation of (updates of) plug-ins by implementation team (Philips (1), VTT (2)) - Subversion (new plug-in) - Doors (new plug-in) - SoftFab (plug-in update) - PAT (plug-in update)
Wk 713	Implementation of tool chain in OSIB project, including real-life project data
Wk 714	Demo to Merlin representatives + Philips project leaders

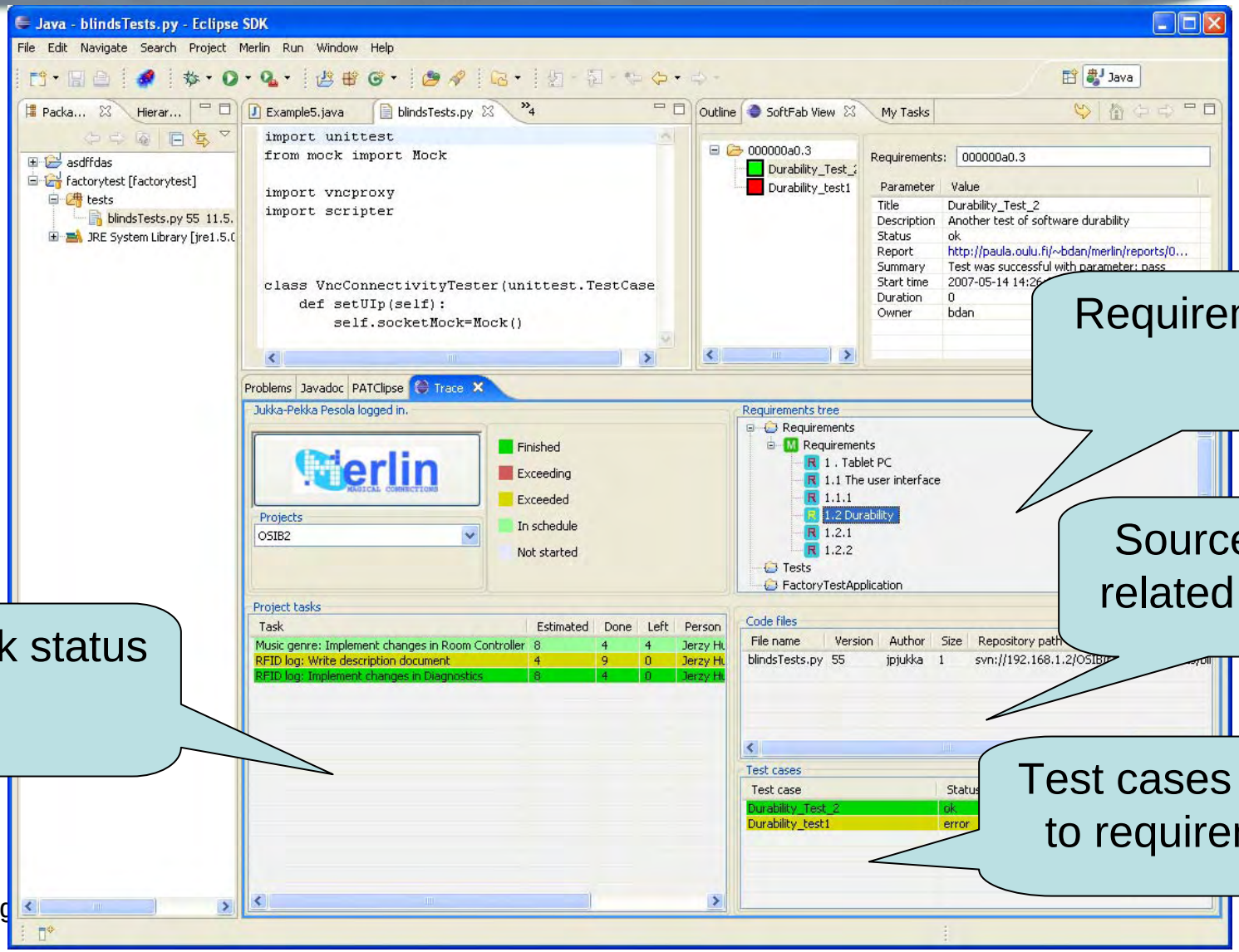
Tool chain screen dumb: Feature-Task Overview



Tool chain screen dumb: Task Burn-down



Traceability Overview



OSIB Conclusions

- Easy preparation of plug-ins
- Fast and successful implementation trajectory
- Full transparency demonstrated on real-life project data, according to expectations
  - Total overview on one screen!
  - Unified user-interface for various views (tasks, requirements, code, build & test)
    - No need to get used to individual tool user-interfaces
- Project leaders enthusiastic:
  “When can I have it!”

Overall Conclusions

- Merlin Tool demonstrates, that integration of tools from different vendors is possible, and it answers directly to the identified challenges
  - Efficient use of resources
  - Transparency through partner borders
  - Traceability between the tested product and the requirements
  - Common understanding of the requirements
  - Consistency in engineering tasks

Current and Future Work

- Improving and further developing the Merlin Tool
  - Adding possibility to exchange tools in the area of:
    - Requirements Management (DOORS → OSRMT)
    - Testing (SoftFab → TestLink)
    - Project management (PAT → Open Workbench)

    Focus on incorporation of open source tools

- Status of the toolchain (e.g. PR/CR database) maintained by VTT

- Iterative evaluation
  - Each 2 weeks feedback from Philips- AppTech, by gathering opinions from users
  - Evaluation results sent to VTT
    Minor problems / change requests have been issued related to
    - User Interface
    - Start-up sequence of tool chain
    - Unexpected behavior
      (e.g., missing information in some cases)

- Create a final report about the tool chain development